UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/753,262	01/05/2004	Simon Joshua Waters	1011-67363	6703

7590 04/12/2006

KLARQUIST SPARKMAN, LLP
One World Trade Center, Suite 1600
121 S. W. Salmon Street
Portland, OR 97204

EXAMINER	
LEVIN, NAUM B	

ART UNIT	PAPER NUMBER
2825	

DATE MAILED: 04/12/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/753,262	WATERS ET AL.	
	Examiner	Art Unit	
	Naum B. Levin	2825	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 January 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *41-87* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *41-87* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *05 January 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *01/05/04*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

This office action is in response to application 10/753,262 and preliminary amendment filed on 01/05/2004. Claims 41-87 remain pending in the application.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

1. Claims 41-87 are rejected under 35 U.S.C. 102(e) as being unpatentable by Schaumont et al. (US Patent 6,606,588).

2. As to claims 41, 51, 58, 63, 69, 72, 75, 79 and 84 Schaumont discloses:

(41) A computer readable medium storing computer executable instructions for causing a computer system programmed thereby to perform a method of transforming a programming language specification into a lower-level specification, the method comprising (col.68, ll.5 7-59, Abstract):

accepting a programming language specification (The design flow according to an embodiment of the present invention, as shown in FIG. 1D, starts off with an untimed, floating point C++ system description 101, this description uses data-flow semantics –col.8, ll.20-25), the programming language specification including an interface (The system is described as a network of communicating components –col.8,

ll.25-26; The file sources and sinks define operating system interfaces- col.16, ll.21-22) (col.8, ll.20-26; col.15, ll.21-33; col.16, ll.19-59); and

based upon a set of transformation rules, transforming plural methods of the interface into plural ports of a port map of a lower-level specification for a design unit (First, we consider the construction of an actor. An actor is a subalgorithm out of a dataflow system model. In OCAPI, each actor is defined by one class. As an example of actor definition, we take the diffenc block out of the transmitter. The include file (3.3) defines a class diffenc (line 10-class diffenc public base) that inherits from a base class. This inheritance defines the class under definition as a dataflow actor. The dataflow actor defines a constructor, a run method and a reset method. The run method (line 25) is the method that is called when the actor should be executed. This method takes along parameters that include the name (name), the I/O ports _sym 1, _symb2). Looking at the implementation of run (include file 3.4, line 26), we distinguish a firing rule in lines 29-30. The firing rule expresses that the run method should return whenever there is no data available in the queue. Otherwise, processing continues as described beyond line 32-col.69, ll.45-65. Finally we indicate the output of the code generation process. When an architecture model is constructed, several code generators can be used. OCAPI currently can generate RT-VHDL code directly- col.71, ll.5-8) (col.8, ll.59-61; col.15, ll.54-67; col.16, ll.1-17; col.69, ll.31-67; col.70, ll.1-67; col.71, ll.1-24);

(51) A computer readable medium storing computer executable instructions for causing a computer system programmed thereby to perform a method of transforming a

programming language specification into a lower-level specification, the method comprising (col.68, ll.5 7-59, Abstract):

accepting a programming language specification, the programming language specification including plural calls to plural instances of a unit class, wherein a first call of the plural calls maps to a first instance of the plural instances of the unit class, wherein a second call of the plural calls maps to a second instance of the plural instances of the unit class, and wherein the programming language specification lacks explicit concurrency modeling for the plural instances of the unit class (col.5, ll.24-49; col.8, ll.20-26; col.15, ll.21-33; col.16, ll.19-59; col.40, ll.17-30); and

transforming the programming language specification into a lower-level specification, wherein the transforming includes generating lower-level description for handling concurrent execution of units represented by the plural instances of the unit class in the programming language specification (In a preferred embodiment, said first refining step comprises the steps of determining the input vector lengths of input, output and intermediate signals, determining the amount of parallelism of operations that process input signals under the form of a vector to output signals –col.5, ll.24-30. Said second refining step comprises preferably transforming said tokens from floating point to fixed point- col.5, ll.47-49) (col.8, ll.59-61; col.15, ll.54-67; col.16, ll.1-17; col.40, ll.17-30; col.69, ll.31-67; col.70, ll.1-67; col.71, ll.1-24);

(58) A design tool comprising (col.8, ll.27-44):

a design input module for accepting an algorithmic specification, the algorithmic specification including plural unit calls that map to plural different instances of a unit,

thereby indicating parallel execution of the plural unit calls (col.5, ll.24-49; col.8, ll.20-26; col.15, ll.21-33; col.16, ll.19-59; col.40, ll.17-30); and

a hardware description language transformer for transforming the algorithmic specification into a lower-level specification, wherein the transformer adds code into the lower-level specification for handling the parallel execution of the plural unit calls (The constructor passes the name of the object to the "base" class it inherits from. In addition, it initializes private members with the other parameters. In this example, the communication queue pointers are initialized. This is not done through simple pointer assignment, but through function calls "asSource" and "asSink". This is not obligatory, but allows OCAPI to analyze the connectivity in between the basic blocks. Since a queue is intended for point-to-point communication, it is an error to use a queue as input or ouput more then once. The function calls "asSource" and "asSink" keep track of which blocks source/sink which queues. The constructor can optionally also be used to perform initialization of other private data (state for instance) –col.15, ll.54-67; In a preferred embodiment, said first refining step comprises the steps of determining the input vector lengths of input, output and intermediate signals, determining the amount of parallelism of operations that process input signals under the form of a vector to output signals, determination of objects, connections between objects and signals between objects of said data-flow model, and determining the wordlength of said signals between objects- col.5, ll.23-30) (col.8, ll.59-61; col.15, ll.54-67; col.16, ll.1-17; col.40, ll.17-30; col.69, ll.31-67; col.70, ll.1-67; col.71, ll.1-24);

(63) A design tool comprising col.8, ll.27-44):

one or more modules for accepting an object-oriented description, the object-oriented description including an interface (col.5, ll.24-49; col.8, ll.20-26; col.15, ll.21-33; col.16, ll.19-59; col.40, ll.17-30); and

one or more modules for translating the object-oriented description into a lower-level specification having plural ports specified according to defined semantics for the interface of the object-oriented description (col.8, ll.59-61; col.15, ll.54-67; col.16, ll.1-17; col.40, ll.17-30; col.69, ll.31-67; col.70, ll.1-67; col.71, ll.1-24);

(69) In a computing environment, a computer-implemented method of transforming a programming language specification into a lower-level specification, the method comprising (col.68, ll.5 7-59, Abstract):

providing a programming language specification, the programming language specification including an interface (col.8, ll.20-26; col.15, ll.21-33; col.16, ll.19-59); and

receiving a lower-level specification produced by transforming the programming language specification into the lower-level specification, the lower-level specification having plural ports specified according to defined semantics for the interface of the programming language specification (col.8, ll.59-67; col.15, ll.54-67; col.16, ll.1-17; col.40, ll.17-30; col.69, ll.31-67; col.70, ll.1-67; col.71, ll.1-24);

(72) In a computing environment, a computer-implemented method of transforming a programming language specification into a lower-level specification, the method comprising (col.68, ll.5 7-59, Abstract):

providing a programming language specification, the programming language specification including plural unit calls that map to plural different instances of a unit class, thereby indicating parallel execution of the plural unit calls col.5, ll.24-49; col.8, ll.20-26; col.15, ll.21-33; col.16, ll.19-59; col.40, ll.17-30); and

receiving a lower-level specification produced by transforming the programming language specification into the lower-level specification, wherein the transforming includes adding code into the lower-level specification for handling the parallel execution of the plural unit calls (col.8, ll.59-61; col.15, ll.54-67; col.16, ll.1-17; col.40, ll.17-30; col.69, ll.31-67; col.70, ll.1-67; col.71, ll.1-24);

(75) A computer-readable medium storing computer-executable instructions for causing a computer system programmed thereby to perform a method comprising:

receiving an object-oriented description, the object-oriented description including native programming language code, wherein the object-oriented description specifies structural details of a design unit for synthesis with a design tool into a HDL description of the design unit(col.8, ll.20-26; col.15, ll.21-33; col.16, ll.19-59; col.63, ll.1-13); and

compiling the object-oriented description with a native programming language compiler for algorithmic validation of the design unit independent of the synthesis (col.8, ll.59-61; col.15, ll.54-67; col.16, ll.1-17; col.63, ll.1-13; col.69, ll.31-67; col.70, ll.1-67; col.71, ll.1-24);

(79) In a design tool, a method for implementing a hierarchical relationship between a first design unit and a second design unit, the method comprising:

automatically creating one or more local signals of a first design unit, the first design unit including an instance of a second design unit without explicitly specifying connection logic between the one or more local signals and corresponding ports of the second design unit calls (col.5, ll.24-49; col.8, ll.20-26; col.15, ll.21-33; col.16, ll.19-59; col.40, ll.17-30); and

automatically mapping the one or more local signals to the corresponding ports of the second design unit (col.8, ll.59-61; col.15, ll.54-67; col.16, ll.1-17; col.40, ll.17-30; col.69, ll.31-67; col.70, ll.1-67; col.71, ll.1-24);

(84) A method of creating a system including software and hardware, the method comprising:

accepting a programming language specification of a system including software and hardware, wherein the programming language specification follows the same syntax for the software and the hardware (The C++ description also illustrates some of the main contributions of OCAPI: register-transfer level aspects (signals, clocks, registers), as well as dataflow aspects simulation queues) are freely intermixed and used as appropriate. By making use of C++ operator overloading and classes, these different design concepts are represented in a compact syntax format -col.58, ll.21-28) (col.8, ll.20-26; col.15, ll.21-33; col.16, ll.19-59); and

transforming at least part of the programming language specification into a lower-level specification(col.8, ll.59-61; col.15, ll.54-67; col.16, ll.1-17; col.69, ll.31-67; col.70, ll.1-67; col.71, ll.1-24).

3. As to claims 42-50, 52-57, 59-62, 64-68, 70-71, 73-74, 76-78, 80-83 and 85-87 Schaumont recites:

(42), (52), (53) The computer readable medium, wherein the method further comprises transforming into a synchronized process of the lower-level specification (col.21, ll.45-54);

(43), (45), (54), (64), (70), (73), (82) The computer readable medium/tool/method, wherein transforming includes transforming each input/output method into an input/output port of the port map (col.8, ll.59-61; col.15, ll.54-67; col.16, ll.1-17; col.69, ll.31-67; col.70, ll.1-67; col.71, ll.1-24);

(44) The computer readable medium, wherein a method of the interface includes as a parameter a pointer to a shared variable, wherein the shared variable is for modeling in/out behavior (col.17, ll.63-67; col.18, ll.1-23);

(46) The computer readable medium, wherein the programming language specification includes one or more values specified in a template (col.14, ll.38-55);

(47) The computer readable medium, wherein the programming language specification is for a system including hardware and software (col.6, ll.18-23);

(48) The computer readable medium, wherein the programming language specification includes a C++ class description (col.60, ll.16-24);

(49), (56), (61), (67), (81), (86) A file including the programming language specification of claim 41 (col.17, ll.39-52);

(50), (57), (62), (68), (78), (83), (87) A file including the lower-level specification of claim 41 (col.36, ll.63-67; col.37, ll.1-18);

(55) The computer readable medium/tool, wherein the programming language specification includes an object-oriented class description (col.63, ll.1-13);

(59), (66) The design tool further comprising an architecture exploration module for presenting alterative architectures (col.69, ll.30-38);

(60) The design tool, wherein the unit is a sub-design unit (col.56, ll.8-33);

(65) The design tool, wherein the object-oriented description further comprises a private interface (col.14, ll.56-67; col.15, ll.1-33);

(71), (74) The method, wherein the computing environment is a distributed computing environment (col.50, ll.21-34);

(76) The computer-readable medium of claim 75 wherein the native programming language is C++ (col.7, ll.20-59);

(77), (80) The computer-readable medium of claim 75, wherein the method further comprises performing algorithmic simulation (col.56, ll.6-33);

(85) The method further comprising simulating the transformed lower-level specification (col.8, ll.52-57).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Naum B. Levin whose telephone number is 571-272-1898. The examiner can normally be reached on M-F (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jack Chiang can be reached on 571-272-7483. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

N L

THUAN DO
Primary examiner
4/07/2006